# SmartKeepers: a decentralized, secure, and flexible social platform for coworkers

Romain Blin[1], Charline Berthot[1], Julien Subercaze[2], Christophe Gravier[2], Frederique Laforest[2], and Antoine Boutet[2]

[1] smart keepers, Saint-Etienne, France
firstname.lastname@univ-st-etienne.fr
[2] CNRS, Laboratoire Hubert Curien, Saint-Etienne, France
firstname.lastname@smart keepers.com

**Abstract.** Coworking has emerged as an attractive model for organization. It relies on a highly dynamic collaborations among different partners. Traditional centralized social platform lack fundamental requirements for such collaborations potentially short, that are the management of the dynamic topology of such professional networks, privacy, data exchanges and ownership. In this paper, we present SmartKeepers, a decentralized and secure environment for coworking activity. Each user physically owns her node that she plugs in and out as she moves from one collaborative space to another. The system supports a large variety of network topologies and is fully interoperable with W3C compliant solutions. We showcase the cogency of the Semantic Web for building decentralized and secure services while keeping every user at the core of the data ownership process.

## 1 Introduction

Coworking has emerged as an attractive model for organizations [4]. The rise of coworking spaces in the past decade has brought new challenges for adapted collaborative social platforms. Coworking spaces offer supports for remote working, freelance and entrepreneurial activities. Coworkers share physical resources such as desks, Internet accesses, printers, conference rooms or kitchen, to name a few. The dynamic and collaborative nature of coworking pushes freelancers, entrepreneurs and independent contractors to collaborate and start new projects. As a consequence, individuals might belong to several projects or organizations such as bootstrapping a startup with a few partners and managing a freelance activity at the same time. The same individuals may integrate two other collaborative projects several weeks later. To follow this dynamics, digital assets management requires flexibility, fine-grained control and a high level of security.

Traditional enterprise social network services such as Yammer [3] are originally designed to manage large companies. They are therefore not adapted for the paradigm shift introduced by coworking [5]. Moreover, using well known social networks or a myriad of different public solutions for a business activity most likely represent a risky solution in case of a leak of privacy [1]. The usage of such solutions can be also prohibited in case of confidential documents which must not be disseminated outside the

[3] https://www.yammer.com

partners. In the last few years, decentralization has become an appealing solution to address the challenges of privacy and data ownership [2,3,6].

In this demonstration we present SmartKeepers, a cost-effective and scalable decentralized social network solution for coworkers looking for flexibility. SmartKeepers addresses two main challenges raised by the organizational revolution introduced by coworking: decentralization and security. SmartKeepers offers all the required tools for daily IT activities of every business such as shared calendars, collaborative documents, real-time discussion, backup or communication services. SmartKeepers leverages the elastic nature of decentralized solutions to be cost-effective and flexible, adapting the infrastructure to the need of dynamic companies. The solution is based on W3C standards at every level of its architecture. It relies on Semantic Web standards for data representation and exchange. It implements state-of-the-art cryptographic protocols and standards to ensure security. Fine-grained access control is realized through Web Access Control and WebID. Based on a decentralized model, the SmartKeepers system allows users to regain complete ownership and control over their data.

The lightweight version of SmartKeepers is bundled on a low-consumption device (the Raspberry Pi in this demonstration) and is called KeepBox. This allows any user to host his own data for low starting and running costs. Different coworkers' boxes can interact with each other to offer shared spaces and communication functionalities while respecting privacy using fine-grained security policies. In this demonstration we present the capabilities of SmartKeepers to operate with one KeepBox as well as with multiple KeepBoxes to scale the infrastructure of a growing business. It also illustrates how the shared collaborative tools work between different partners.

## 2 Decentralized openness

SmartKeepers is designed to support the full range of real life situations that emerge from the paradigm shift introduced by coworking. For this purpose, the network architecture of SmartKeepers is highly flexible and supports various network topologies. Communication between users is supported by the communication between the KeepBoxes. Each user is identified by the URI of his FOAF profile. Each user is hosted on a single box. However a box can host several users. This model enables the decoupling from the user and the hosting of her data. SmartKeepers grants elasticity in two ways:

**User relation:** Relations between users evolve over time. This is a standard feature of social networks. In traditional centralized networks such a Facebook, this relation is represented by a value in a database. In SmartKeepers this relation is represented through cryptographic certificate authorization (See Section 3.1).

**Data migration:** SmartKeepers offers the users to migrate their data from one KeepBox to another. For instance, a user who is leaving a company to start as freelancer can transfer his data from the company box to a privately hosted box. A user who is unsatisfied by the quality of an hosting service, may decide to migrate her data to another one.

For interoperability purposes, SmartKeepers is fully committed to support W3C standards in every piece of its architecture: security, data, protocols. In SmartKeepers, the following standards are implemented in different parts of the architecture:

**Protocols:** To expose data on the Web, SmartKeepers is using the Read Write Web protocol (RWW) over HTTPS. RWW defines performatives to access and modify data over the Web. SmartKeepers utilizes notifications in order to keep users updated of the news from their network. Notifications are implemented upon the Pingback protocol and its related ontology.

**Data:** Every service of SmartKeepers, such as calendar, group management, message board, file exchange, has an underlying data format defined by a standard ontology. FOAF is used to describe a person. Message stream uses the SIOC ontology to describe users' posts and comments. Calendar is based upon NCAL[4], contact management is backed by NCO[5].

**Security:** Security is a key service of SmartKeepers. WebID and Web Access Control are the two standards implemented to harness security. We detail the usage of these standards to implement secure services in the next section.

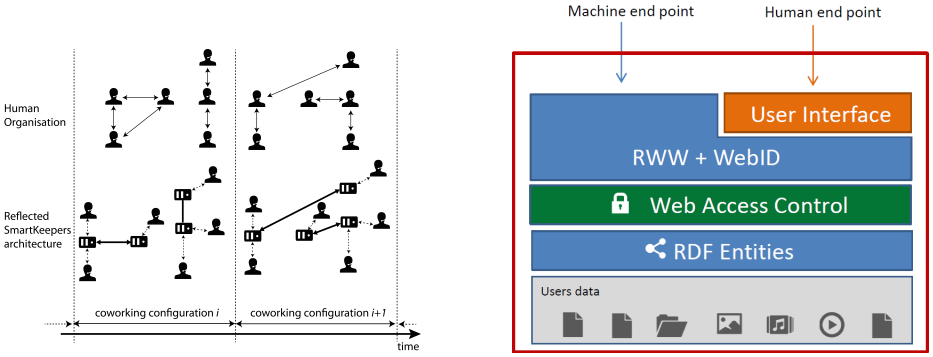


Fig. 1: Example of evolving human co-working organisation (left) and the SmartKeepers stack that supports such configurations (right)

## 3 Secure Services

### 3.1 WebID

The authentication in SmartKeepers relies on WebID over TLS sessions. This is a paradigm under consideration as a W3C specification[6] that allows distributed authentication. In our system, a WebID is an URI which denotes a person. When Bob wants to access Alice's data on her box in order to post new triples, Bob needs to authenticate on Alice's box. A preliminary is for Bob to create a certificate store within its Web browser for authentication purposes. Both private and public keys are stored within Bob's in-browser keystore. This step is usually performed on Bob's KeepBox that Bob controls. Then, when Bob accesses Alice's services, his browser first initiates a TLS

[4] Nepomuk Calendar Ontology
[5] Nepomuk Contact Ontology
[6] `http://www.w3.org/2005/Incubator/webid/spec/tls/`

connection with Alice's KeepBox. Then, the latter issues a `CertificateRequest` message, a performative already present in TLS protocol. Bob's browser provides the certificate without any user interaction, and the authentication scheme is run on Alice's KeepBox. For this purpose, the WebID provided at sign-in is dereferencable using the specified protocol in this URI. Then, Alice's KeepBox checks if the returned profile relates the WebID to the certificate public key. When this authentication process successfully terminates, Bob is connected to Alice's KeepBox, without having to create an account on it. Bob can synchronize his WebIDs on different browsers for every terminal he owns. This is of the utmost practical interest in the SmartKeepers distributed system – authentication is performed without Bob having to either create an account, or use a login/password pair as the certificate is stored in Bob's browser and served automatically on `CertificateRequest` messages.

### 3.2 Web Access Control

Linking WebID with other ontologies natively supports privacy settings for the user controlling his identity on the Web. This is similar to common user and role based access control policies in most Web systems, except that each user is authenticated using an URI. On her KeepBox, Alice manages the access control list of users allowed to access her services (calendar, social stream, posts, notifications, ...). This access control list is a sequence of triples that explicits the rights given to a user $u$ identified by his URI to operate Alice's services. The supported operations are the performatives from RWW for trusted read and write operations. This includes Read, Write, and Append triples to existing RDF graph for each Alice's service. At this stage, it is important to note that the profile of a user $u \in U$ stays on the server of $u$. The information about $u$ is not duplicated, which leaves out any profile synchronicity issue. Moreover, this enforces a government of $u$ profile by $u$ only.

## 4 Demonstration of the system

Our demonstration is a Web-based server implemented in Java [7]. As the protocols implemented in the system are particularly lightweight, the system itself can run on a low cost and credit-card sized computer – the Raspberry Pi, which needs only needs 3.5 Watts per hour.

The unifying thread throughout our demonstration involves two KeepBoxes at the conference site and it evolves around the following scenario. The first KeepBox is configured and hosts the user profile and services of one of the authors – denoted Alice in what follows. The second KeepBox is reset to factory default for each presentation, so that we can showcase the zero-configuration deployment of a new node in the decentralized social network. Other already running KeepBoxes are also connected to the Internet – this represents as many additional nodes to connect to in this demonstration. This central thread holds together the showcase of three different aspects of the SmartKeepers system as follows.

[7] A screeencast is available at `http://smartkeepers.com/files/screencast.ogv`

**Signing up and adding friends:** At the first stage, we showcase the creation of a new user named Paul on a KeepBox. This includes the browser-side generation and storage of the user certificate. Paul is able to use enterprise-ready services (calendar, mail, ...) but also to search for new friends in the decentralized on-site architecture. He ultimately adds Alice (on-site box) and Carol (remote box) as friends.

**Collaborative agenda and pingbacks:** Users then enter a collaborative task with their friends. Paul suggest a meeting with both Alice and Carol, which are notified using the pingback ontology and its service. In their calendar module, Alice, Paul and Carol view the newly created event.

**Meeting follow-up online discussion and file sharing:** Right after the meeting, Paul wants to share his meeting minutes with Alice and Carol. He uploads his minutes through the file manager module, and he is able to share it with Alice and Carol only if he edits his web access control list in order to grant them the right to access the file on his box. This illustrates the RWW, WebID and Web Access Control stack, all implemented using TLS sessions.

## 5 Conclusion

We demonstrate the cogency of the Semantic Web for building an enterprise-ready decentralized and secured social platform. The SmartKeepers system makes it possible for any user to wield control of his identity in the social network, as well as manage the access control list for the different social services she exposes to her peers. The demonstration showcases services including WebID sign-in on the box of a peer using TLS sessions and certificates – without the need to memorize a login and a password per peer –, collaborative distributed social services based on the pingback ontology for notifications, and Web Access Control ontology for managing peers rights. The SmartKeepers system is also easy to deploy and run on commodity boxes – like the RaspberryPI –, roughly a hundred times less energy consuming than a refrigerator.

## References


1. D. B. Giffin, A. Levy, D. Stefan, D. Terei, D. Mazières, J. C. Mitchell, and A. Russo. Hails: Protecting data privacy in untrusted web applications. In *Operating Systems Design and Implementation*, 2012.
2. D. Koll, J. Li, and X. Fu. Soup: An online social network by the people, for the people. In *Middleware*, 2014.
3. S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam. Prpl: A decentralized social networking infrastructure. In *Mobile Cloud Computing*, 2010.
4. C. Spinuzzi. Working alone together, coworking as emergent collaborative activity. *Journal of Business and Technical Communication*, 26(4):399–441, October 2012.
5. S. Townsend. What are you working on?: Global knowledge sharing at deloitte. *eLearn*, 2012(1), 2012.
6. T. Xu, Y. Chen, J. Zhao, and X. Fu. Cuckoo: Towards decentralized, socio-aware online microblogging services and data measurements. In *Hot Topics in Planet-scale Measurement*, 2010.